http://www.cplusplus.com/reference/

C++刷题常用容器总结

0. quick search

```
vector, 变长数组，倍增的思想
    size()  返回元素个数
    empty()  返回是否为空
    clear()  清空
    front()/back()
    push_back()/pop_back()
    begin()/end()
    []
    支持比较运算，按字典序

pair<int, int>
    first, 第一个元素
    second, 第二个元素
    支持比较运算，以first为第一关键字，以second为第二关键字（字典序）

string，字符串
    size()/length()  返回字符串长度
    empty()
    clear()
    substr(起始下标，(子串长度))  返回子串
    c_str()  返回字符串所在字符数组的起始地址

queue, 队列
    size()
    empty()
    push()  向队尾插入一个元素
    front()  返回队头元素
    back()  返回队尾元素
    pop()  弹出队头元素

priority_queue, 优先队列，默认是大根堆
    size()
    empty()
    push()  插入一个元素
    top()  返回堆顶元素
    pop()  弹出堆顶元素
    定义成小根堆的方式：priority_queue<int, vector<int>, greater<int>> q;

stack, 栈
    size()
    empty()
    push()  向栈顶插入一个元素
    top()  返回栈顶元素
    pop()  弹出栈顶元素

deque, 双端队列
    size()
```

```
    empty()
    clear()
    front()/back()
    push_back()/pop_back()
    push_front()/pop_front()
    begin()/end()
    []

set, map, multiset, multimap, 基于平衡二叉树（红黑树），动态维护有序序列
    size()
    empty()
    clear()
    begin()/end()
    ++, -- 返回前驱和后继，时间复杂度 O(logn)

    set/multiset
        insert()  插入一个数
        find()  查找一个数
        count()  返回某一个数的个数
        erase()
            (1) 输入是一个数x，删除所有x   O(k + logn)
            (2) 输入一个迭代器，删除这个迭代器
        lower_bound()/upper_bound()
            lower_bound(x)  返回大于等于x的最小的数的迭代器
            upper_bound(x)  返回大于x的最小的数的迭代器
    map/multimap
        insert()  插入的数是一个pair
        erase()  输入的参数是pair或者迭代器
        find()
        []  注意multimap不支持此操作。 时间复杂度是 O(logn)
        lower_bound()/upper_bound()

unordered_set, unordered_map, unordered_multiset, unordered_multimap, 哈希表
    和上面类似，增删改查的时间复杂度是 O(1)
    不支持 lower_bound()/upper_bound()， 迭代器的++, --

bitset, 压位
    bitset<10000> s;
    ~, &, |, ^
    >>, <<
    ==, !=
    []

    count()  返回有多少个1

    any()  判断是否至少有一个1
    none()  判断是否全为0

    set()  把所有位置成1
    set(k, v)  将第k位变成v
    reset()  把所有位变成0
    flip()  等价于~
    flip(k) 把第k位取反
```

1. vector

动态数组，方便动态扩容，方便的变量初始化(int类型默认初始化为0，bool默认初始化为false)，可以用来实现邻接表（结点数太多的图）。

头文件

```
#include<vector>
using namespace std;
```

定义

```
//typename可以是基本数据类型/其它标准STL容器/自定义结构体
vector<typename>name;
vector<int>v1;
vector<vector<it> >v2;//两个维度都是动态的
vector<vector<int> > newOne(r, vector<int>(c, 0));//初始化二维vector
vector<student> v3(10);//一个固定为10，二位动态
```

元素访问

```
//1.下标访问
v[i];
//2.迭代器访问
vector<typename>::iterator it;

//另一种迭代器简易定义方法
for(auto it=v.begin();it!v.end();it++) cout<<*it;
//迭代器it可以进行算术运算
```

常用函数

函数	说明
push_back(x)	将元素x添加到容器末尾
emplace_back(x)	将元素x添加到容器末尾，速度快
pop_back()	删除容器末尾元素
size()	获得容器大小
clear()	清空元素
insert(it,x)	在it处插入一个元素x
erase(it)	删除it处的元素
erase(first,last)	删除[first,last)区间内的元素

使用场景

- 元素个数不确定
- 用于实现邻接表存储图

2. set

内部自动有序且不含重复元素的集合
头文件

```
#include<set>
using namespace std;
```

定义

```
set<typename> name;
```

元素访问

```
for(auto it=v.begin();it!=v.end();it++) cout<<*it;
```

常用函数

函数	说明
insert(x)	将元素x插入set容器中，并自动递增排序和去重
find(value)	查找值为value的元素，返回对应迭代器
erase(value)	删除值为value的元素
erase(first,last)	删除[first,last)区间内的元素
size()	返回容器大小
clear()	清空容器

使用场景

- 自动去重并按升序排序
  扩展
- 需要元素不唯一，使用multiset
- 需要元素不排序，unordered_set（内部以散列代替了set内部的红黑树）

3. string

字符串，可以替换为C语言版本的字符数组 char*
头文件

```
#include<string>
using namespace std;
```

定义

```
string str;
string s1("Hello");
string s2(5,'*');           //s2 = "*****"
```

内容访问

```
///1.下标访问
str[i]

//2.迭代器访问
for(auto it=str.begin();it!str.end();it++) cout<<*it;

//3.输出输出只能用cin和cout，除非转化为字符数组
cin >> str;
cout << str;
```

常用函数

函数	说明
operator+=	字符串拼接
compare operator	按照字典序列比较大小
length()/size()	获得字符串大小
insert(pos,str)	pos是int类型，表示在特定位置插入str
insert(it,first,last)	在it出插入[first,last)范围内的字符串
erase(it)	删除it处的元素
erase(first,last)	删除[first,last)范围内的元素
erase(pos,length)	删除pos位置开始的length个字符
clear()	清空字符串
substr(pos,len)	返回从pos位置开始len长度的字符串
string::npos	作为find函数未找到的判断依据
find(str)	返回str第一次出现的位置
replace(pos,len,str2)	从pos位置开始，长度为len的子串替换为str2

4. map

将任何基本类型映射到任何基本类型（包括STL容器）
可以用于hash散列（当元素个数比较多时，不适合用数组散列，就用map）
头文件及定义

```
#include<map>
using namespace std;
map<typename1,typename2> mp;
```

元素访问

```
//1.下标
mp['c']    mp[0]   mp["key"]
//2.迭代器
for(auto it=mp.begin();it!=mp.end();it++)
    cout<<it->first<<" "<<it->second;
```

常用函数

函数	说明
find(key)	返回键为key的映射的迭代器
erase(it)	删除it处的元素
erase(key)	删除key
erase(first,last)	删除[first,last)区间内的元素
size()	返回容器大小
clear	清空容器
count(key)	key存在返回1，否则返回0

使用场景

- 散列表
- 其他映射
  扩展
- 多映射，即一个key对应多个value，使用mutimap
- unordered_map可以替代map,内部实现使用散列代替了map内部的红黑树实现，用于处理只映射而不需要按key来排序的需求，速度快

5. queue

队列，先进先出的容器，常用语广度优先遍历BFS
头文件及定义

```
#include<queue>
using namespace std;
queue<typename>name;
```

常用函数

函数	说明
push(x)	元素x入队列
front()	访问队列的首元素
back()	访问队列的尾元素
pop()	队首元素出队
empty()	检测是否为空队列
size()	队列大小

使用场景

- BFS
  另外，需要注意使用front(),back(),pop()函数前，必须判断是否为空队列（empty函数）
  扩展
- deque(double end queue,双端队列)首位皆可插入和删除u
- priority_queue,使用堆实现的默认将当前队列最大元素置于队首的容器

6. priority_queue

优先队列，默认情况下是将队列中最大元素置于队首，优先级可以自定义。每次进行push().pop()操作，底层的数据结构堆（heap）都会随时调整调整，使优先级最高的元素永远在队首。
头文件

```
#include<queue>
using namespace std;
priority_queue<typename> name;
```

常用函数

函数	说明
push(x)	将元素x入队
top()	访问队首元素
pop()	将队首元素出队
empty()	检测是否为空队列
size()	返回队列大小

优先级设定
基本数据类型(int , double, char)默认数字大或字典序大的优先级高。

```
priority_queue<int,vector<int>,greater<int>> q;
```

- greater表示这个数字越小，优先级越高
- less表示数字越大，优先级越高（默认情况）
  vector表示底层数据堆(heap)的容器

结构体优先级设置

```
#include<iostream>
#include<string>
#include<queue>
using namespace std;
struct fruit{
    string name;
    int price;
    friend bool operator < (fruit f1,fruit f2);
        return f1.price>f2.price;
    }
}f1,f2,f3;
int main(){
    priority_queue<fruit> q;
    f1.name="桃子";
    f.price=3;
    f1.name="梨子";
    f.price=4;
    f1.name="苹果";
    f.price=1;
    q.push(f1);
    q.push(f2);
    q.push(f3);
    cout<<q.top().name<<" "<<q.top().price<<endl;
    return 0;
}
```

使用场景
可以用于dijkstra算法的顶点选择中
注意使用top函数之前判断队列是否为空（empty()函数）

7. stack

栈，先进后出
头文件及定义

```
#include <stack>
using namespace std;
stack<typename> name;
```

常用函数

函数	说明
push(x)	元素x入栈
top()	获取栈顶元素
pop()	栈顶元素出栈
empty()	检测stack是否为空
size()	获取栈的大小

使用场景

递归模拟，防止递归深度过深，DFS模拟

8. pair

头文件及使用

```
#include<utility>
using namespace std;
//map实现中设计pair，使用map时自动添加了该头文件

pair<typename1,typename2> name;
paif<string,int> p("hahahah",7);
make_pair("haha",4);
cout<<p.first<<" "<<p.second;
```

9. C++区别于C

namespace, cin/cout, 变量声明(for 循环), bool, const, string, 结构体定义(可不加 struct), 引用, auto